Requirements and Analysis Document for the BrawlBuddies project (RAD)

Contents

Version: 2.0

Date: 2014-05-25

Author: David Gustafsson, Matz Larsson, Lisa Lipkin, Patrik Haar

This version overrides all previous versions.

1 Introduction

This section gives a brief overview of the project.

1.1 Purpose of application

We want to create a 2D Platform Fighting game in which you can be a character with attacks/skills and play against another person locally.

1.2 General characteristics of application

The application will be a desktop, standalone (non-networked), multiplayer application with a graphical user interface for the Windows/Mac/Linux platforms.

The application will be a fast paced non-turn based 2D Platform Fighting game where player can jump around on platforms and fight with other players locally.The player can damage other players and can also take damage from other players. The rounds can be time or life dependent, according to the players choice. The game ends when only one player is left alive or the time have run out.
The GUI will incorporate a kind of HUD that shows information about the characters during the game. Players will also be able to choose character and make other choices from a graphical menu before game. The player will be able to watch the result of the game and information about their performance in game.

1.3 Scope of application

Due to the limited time at our disposal we will NOT be including things like:

-Multiplayer over INTERNET
-Account and account progress
-Story
-AI
-Structures

1.4 Objectives and success criteria of the project

1. It should be possible to play a match of the game and win/lose against another player on the same computer. The player should be able to use one of at least two different characters.
2. The game type can be a timed game or end when only one character is left alive.

1.5 Definitions, acronyms and abbreviations

- GUI: Graphical User Interface
- HUD: Heads Up Display
- Match: A game round to decide a winner.
- AI: Artificial Intelligence

2 Requirements

2.1 Functional requirements

The player(s) should be able to:

1. Start a new game
    a) Each player may choose a character.
    b) Choose one game mode and a map for all players.
    c) Each player may choose a input device.
2. Interact in the world
    a) Move around using the chosen input.
    b) Attack either by using a skill.
    c) Be attacked and lose HP if hit.
3. Win/lose a game.

2.2 Non-functional requirements

2.2.1 Usability

It should be fast and easy to start a game (minimum amount of clicks). Due to the game's fast-paced nature it will be geared towards getting a responsive game with fast response times and a movement system that doesn't feel "sluggish" or slow. While in the game the user should not feel a lack of information, all relevant information should always be available in the HUD.

2.2.2 Reliability

NA

2.2.3 Performance

Any action performed by a user in game should not have a lag exceeding 50ms. Any action performed by a user while navigating the menu should not exceed 2sec. Loadtime of a game should be quick enough not to cause distress to user.

2.2.4 Supportability

The application must be implemented so that the GUI can easily be adapted to the internet. No other platform than PC is supported.

The implementation should prepare for the dividing of the application into a client/server-architecture for net based games. It should be easy to partition the application into a client-server architecture.

2.2.5 Implementation

To achieve platform independence the application will use the Java environment. All hosts must have the JRE installed and configured. The application will not need any installation.

2.2.6 Packaging and installation

The program will be delivered as executable jar file and a README with instructions.

2.2.7 Legal

There should be no copyright conflicts with the application. All elements will either be original or open source with references.

2.3 Application models

2.3.1 Use case model

See APPENDIX for UML diagram and textual descriptions.

2.3.2 Use cases priority

1. Move left/right
2. Jump/fall
3. Use attack
    a. melee
    b. range
    c. range with diagonal movement
4. Start new game

2.3.3 Domain model

See APPENDIX

There will be unique id's for:

- Players
- Game objects (characters, projectiles etc.)
- Skills

2.3.4 User interface

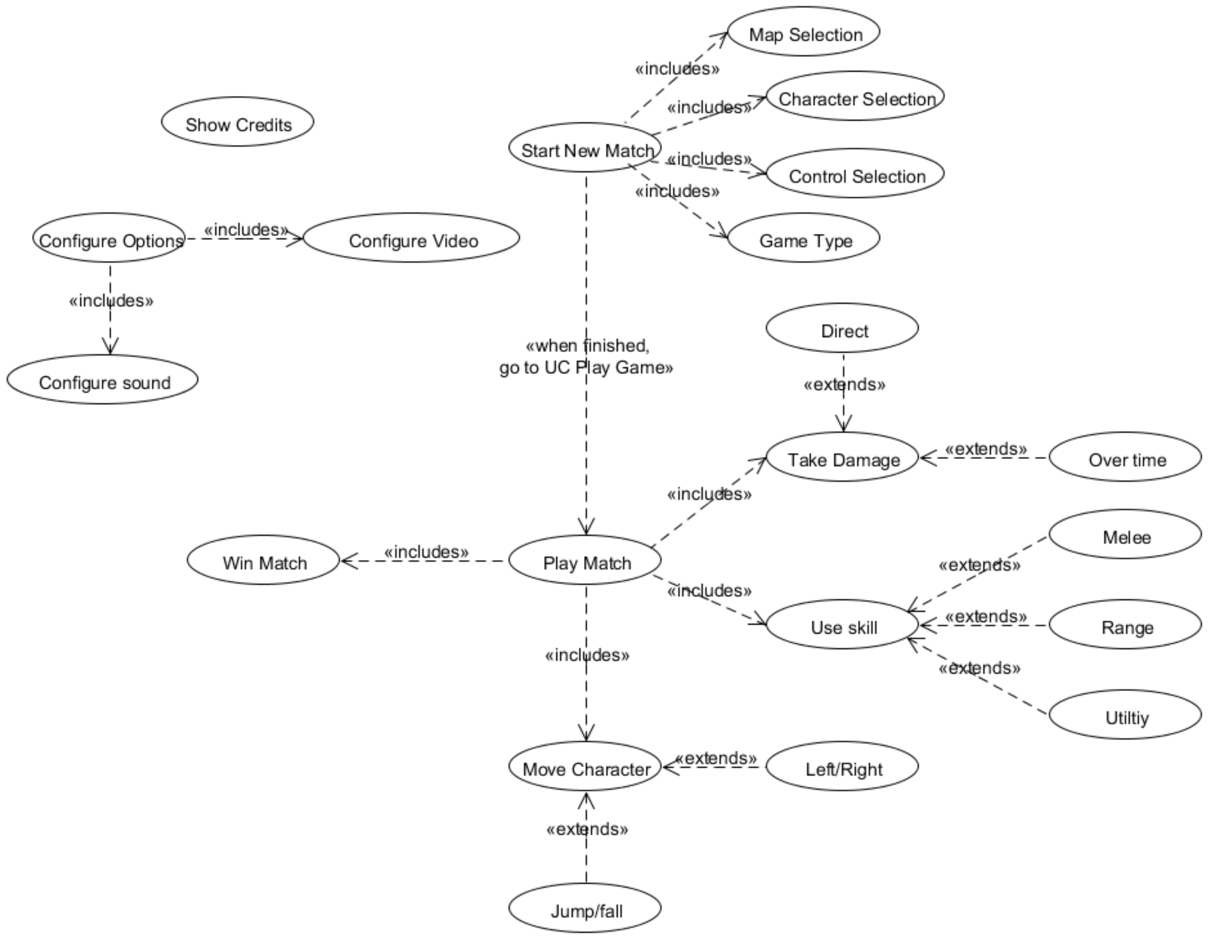
The application should support any resolution of screen although the menus may only allow users to choose among certain values. The user should be able to toggle fullscreen on/off.

2.4 References

No references to declare.

APPENDIX

Use Case overview



Use case texts

Start New Match

Summary: This is how the player moves through the New Match wizard. The application must have been launched before this UC. UC Play Match will follow this UC during normal flow of events.
Priority: medium - high
Extends: -
Includes: Select map, select character, select match type. All three are fully included here.
Participators: Actual player

Normal flow of events

The players choose game mode, characters, names and map. Then the match starts.

	User	System
1	Clicks "Start Game" button	
2		Start new game wizard on step 1, showing character selection, mode selection, name entry and player count
3	Clicks "Time-limit" arrows	
4		Increase/decrease the time-limit within pre-set intervalls
5	Clicks "Life-limit" arrows	
6		Increase/decrease the life-limit from 0 to 10
5	Clicks 1st player character arrows	
6		Cycle through the available characters
7	Clicks 1st player control-input arrows	
8		Cycle through the available inputmethods
	Repeat steps 5-8 for 2nd player	
9	Clicks “Map” arrows	
10		Cycle through the available maps
18	Clicks “Start”	
19		Load characters and map, spawns players and starts match

Move left/right

Summary: This is how the player moves the character in the game. UC Start New Match must have been executed before this UC, though not necessarily right before.
Priority: high
Extends: Move Character
Includes: -
Participators: Actual player

Normal flow of events

A simple move left or right with no consequences.

	User	System
1	Press left/right key.	
2		Character change direction.
3		Character image align with direction.
4		Character move left/right with speed depending on the character specification. The characters position will be updated during next update phase.

Jump

Summary: This is how the player gets the character to jump in the game. UC Start New Match must have been executed before this UC, though not necessarily right before.
Priority: high
Extends: Move Character
Includes: Move-Fall
Participators: Actual player

Normal flow of events

A simple jump straight up with variation if user release the button early or character collides with a solid object.

	User	System
1	Player presses Jump	
2		Character image set to airborne
3		Character gets vertical speed
4		Character is affected by gravity: increasing downwards vertical speed
5		If character hits ground with vertical speed != 0, vertical speed is set to zero.
6		Character image set to idle
5,1		Character collides with impassible object on the way up
5,2		Vertical speed set to zero

Fall

Summary: This is how the player gets the character to fall in the game. UC Start New Match must have been executed before this UC, though not necessarily right before. Player must be in the air, either by jumping or by walking off a ledge.
Priority: high
Extends: Move Character
Includes: -
Participators: Actual player

Normal flow of events

A character is in the air and accelerating towards the ground.

	User	System
1	Player falling while no side-movement.	
2		Character direction and image change to down
3		Character image change to down
4		Character move with gravitational speed . The characters position will be updated during next update phase.

Use Melee/Range Skill

Summary: This is how the player executes a melee/range skill in the game. UC Start New Match must have been executed before this UC, though not necessarily right before.
Priority: high
Extends: Use skill
Includes: Take damage though not explained here.
Participators: Actual player

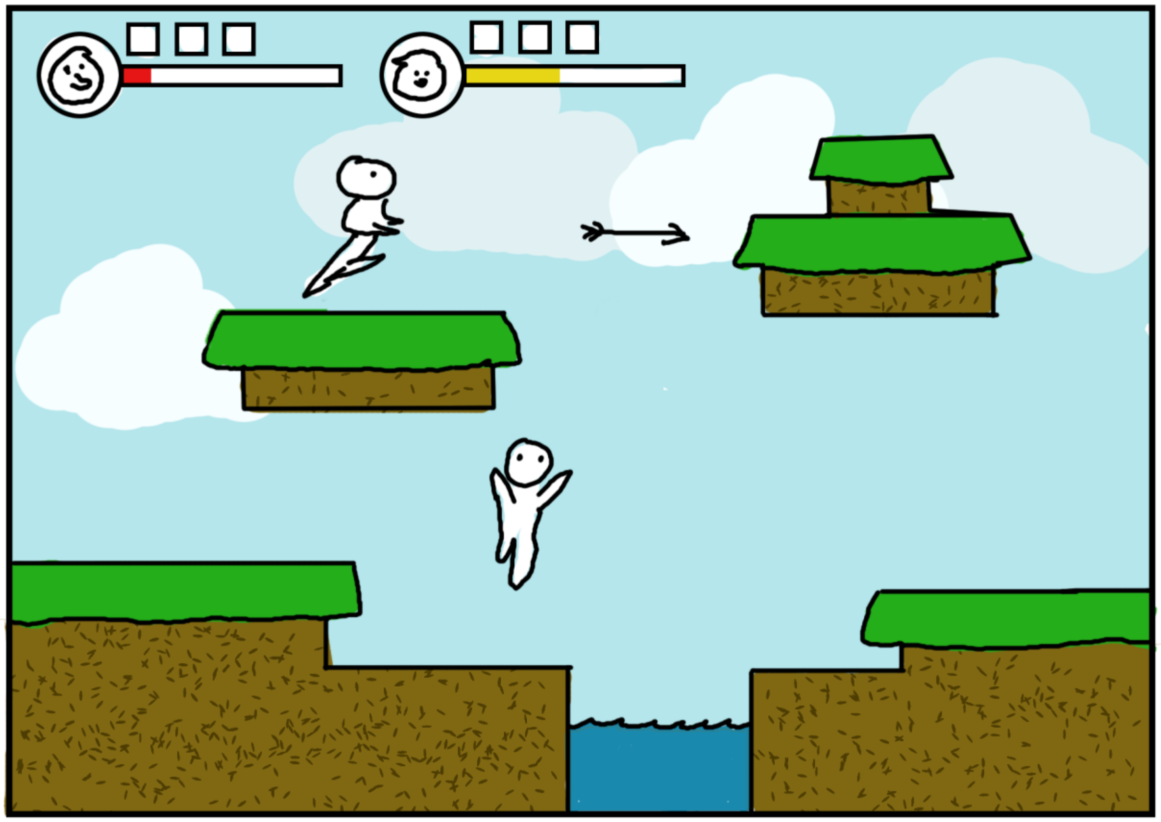
Normal flow of events

Player presses the melee/range attack button and the character performs the attack in the direction the character is currently facing.

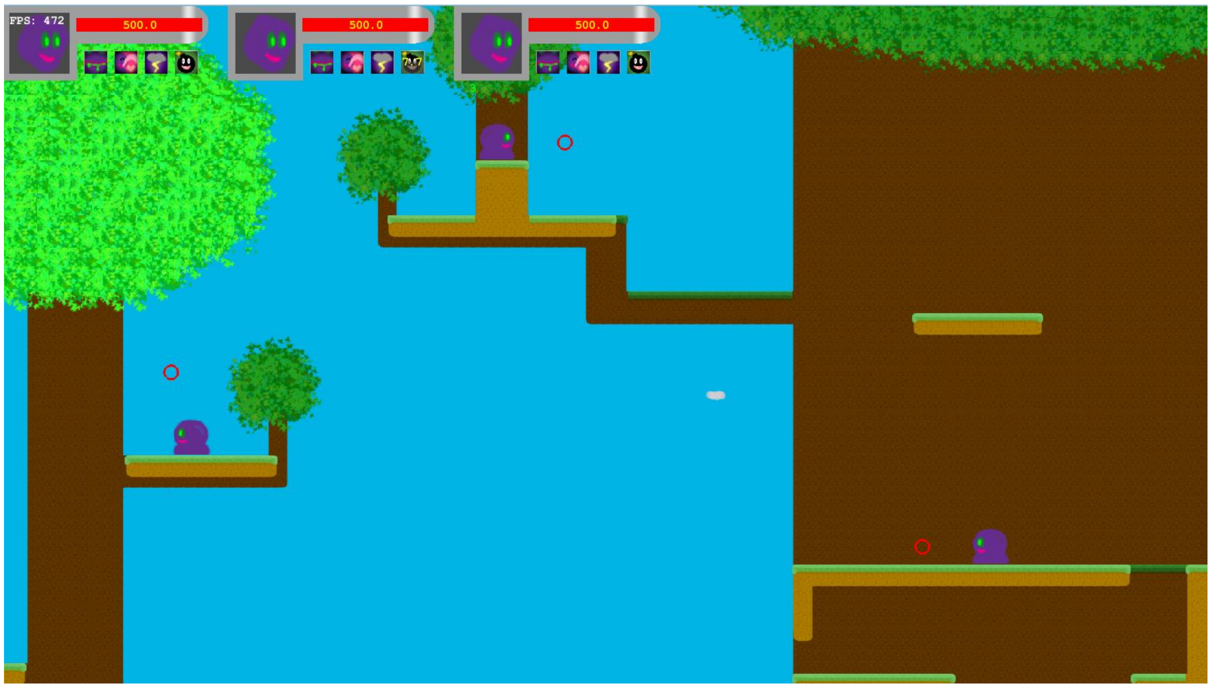
	User	System
1	Player presses Melee/Range Skill key	
2		Player character image change to melee/range skill animation.
3		Player character execute melee/range skill in character direction.
4		Enemy character takes damage if hit.
5		HUD is updated

GUI

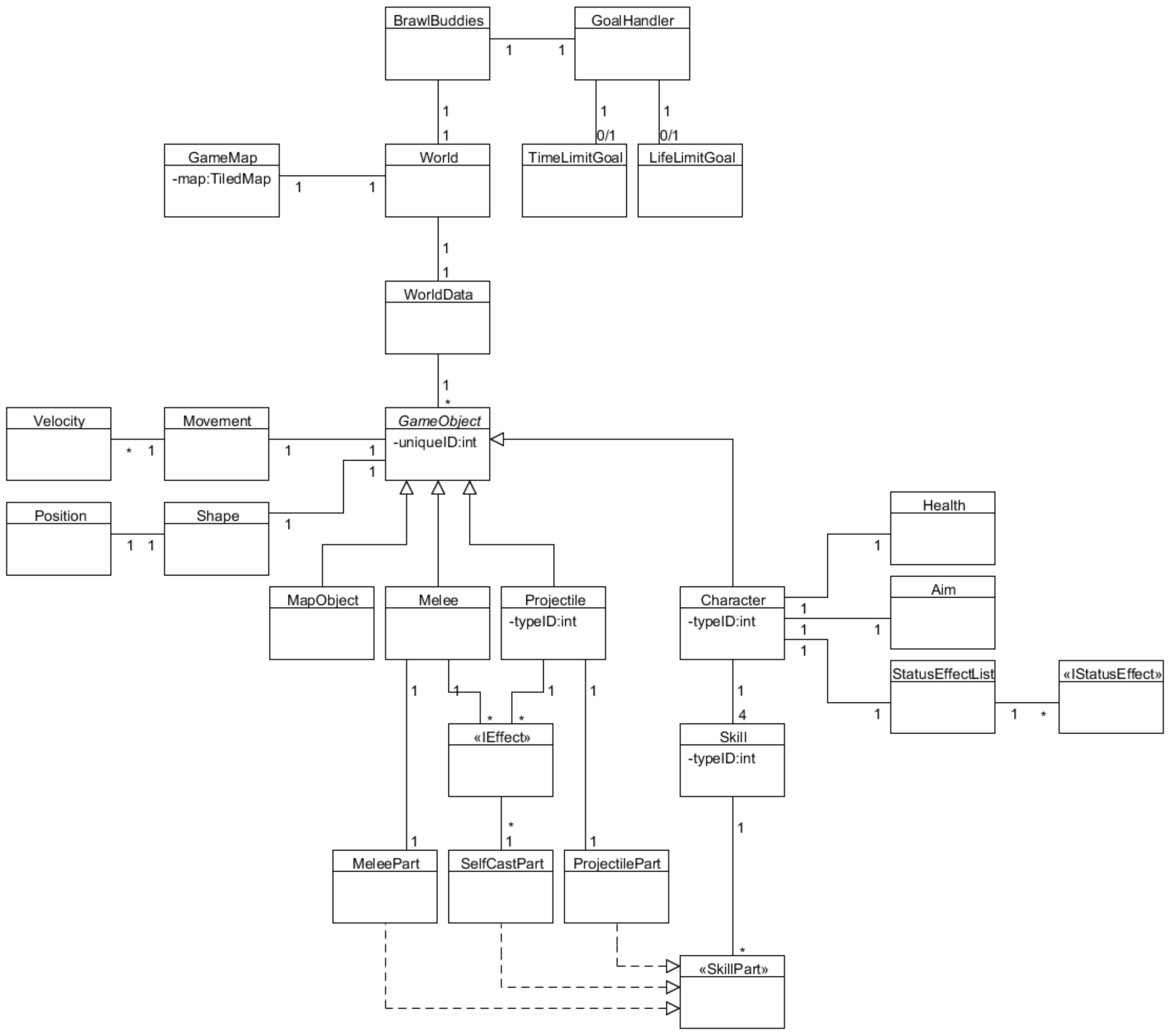
A simple GUI sketch.



The final GUI used by the application.



Domain model


BrawlBuddies
GoalHandler
GameMap
-map:TiledMap
World
TimeLimitGoal
LifeLimitGoal
WorldData
Velocity
Movement
GameObject
-uniqueID:int
Position
Shape
Health
MapObject
Melee
Projectile
-typeID:int
Character
-typeID:int
Aim
StatusEffectList
«IStatusEffect»
«IEffect»
Skill
-typeID:int
MeleePart
SelfCastPart
ProjectilePart
«SkillPart»